Exam 3 CS-GY 6033 INET Fall 2024
December 18th 2024

Instructions:

Scheduling:

- The exam runs from 6pm to 9:00pm on December 18th, 2024. Your exam is run through Gradescope, and will available to download at 5:50pm. The exam is to be completed by 9:00pm. Gradescope stops accepting uploads at 9:15pm. There are absolutely no late uploads accepted by the system after that time. The time to complete the exam will depend on your preparation: a prepared student could finish it in less than two hours, an ill-prepared student might take up to three hours. It is your responsibility to allow time for uploading your exam.

Format:

- The exam consists of a total of 100 points. The written portion is worth 80 points, and your online quiz is worth 20 points.
- **Submission Penalty:** if you do not submit your exam on Gradescope, or do not properly assign your pages, you receive a deduction of 3 points on your exam.
- You may write your solutions directly on the downloaded exam paper, *or* in your own format. You are responsible for providing clear and legible solutions to the problems. Your exam must be resubmitted into Gradescope electronically. Ensure that you know how to quickly upload any handwritten material. This is entirely the student's responsibility. You may assign your pages after the deadline.

Questions during the exam:

- You must stay connected to the ZOOM meeting during the exam. All questions are asked directly using the zoom chat tool.

Rules:

- This exam is a **take-home exam**. You may use **only** the resources from the online class (any material on NYU classes for this course) and any type of calculator (although it is not needed).
- Your work must be entirely your own. It is **forbidden to discuss any work with *any* other person**. Furthermore, your work must be done without using internet searches (although this is completely unhelpful for this exam). Any breach of academic honesty will be handled in accordance with the *Student Code of Conduct*, (a copy of which is provided), and in this particular case, taken very seriously.
- You are asked to **read** the attached Student Code of Conduct Section III subsections A,B,C,D,E and **sign** below to acknowledge that you aware of the policy. Once signed, a copy of this page must be uploaded with your exam.

I acknowledge that my submitted Exam work is entirely my own. I have read and am in accordance with the Student Code of Conduct policy of NYU Tandon and fully accept the consequences of breaching the above instructions.

Name: _____

Signature: _____

Instructions

Read the following questions carefully. Recall that you may use algorithms from class. For example, you may reference BFS, Dijkstra's algorthtm, etc, as long as you carefully describe the input parameters.

ZOOM LINK: (microphones off)

REMINDER of SUBMISSION PENALTY: if you do not submit your exam on Gradescope, or do not properly assign your pages, you receive a deduction of 3 points on your exam.

Question 1

8 points

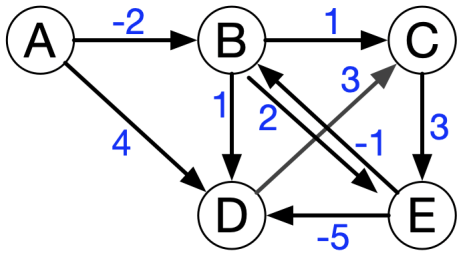
G is a directed, weighted graph (shown below).

- Execute Bellman-Ford's algorithm (to completion) using vertex A as the source, where the edges are processed in the following order

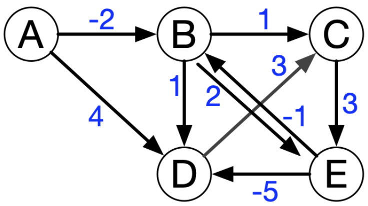
$$AB, AD, BC, BD, BE, CE, DC, EB, ED$$

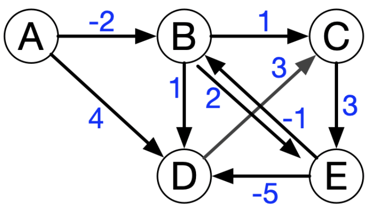
You must show how when a distance attribute and parent attribute are updated, and list when any edges result in no changes. **You must show the current edges of the SSSP tree**.

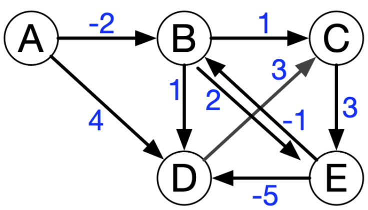
**you may use the next page if you like, to trace out the steps of BF on the graph*

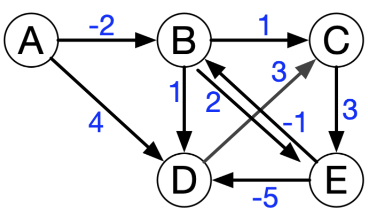


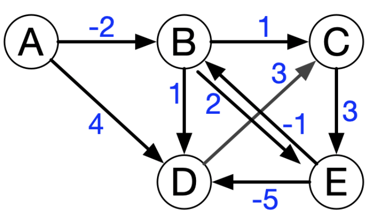
Use this sheet if you like. If you use this sheet, you must clearly label your steps and show your distance attributes as they are updated.

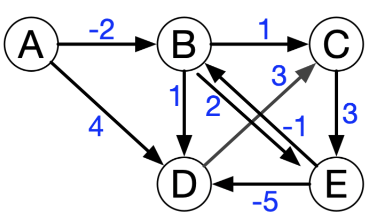


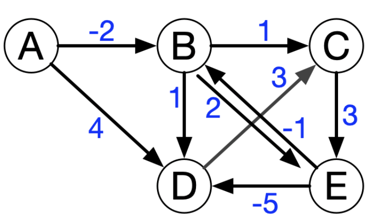


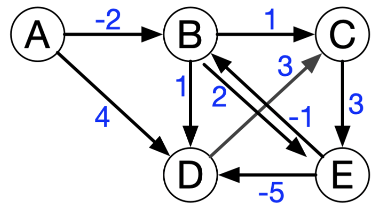


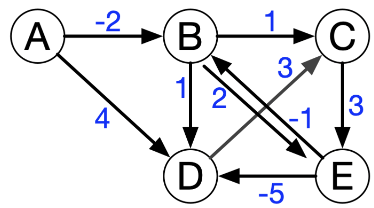


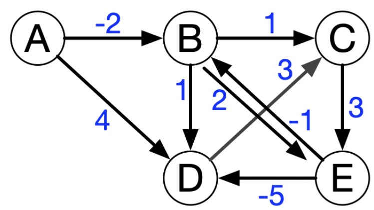


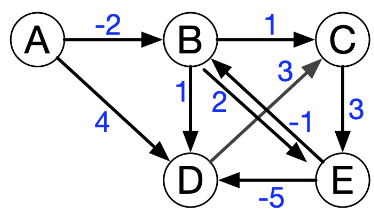


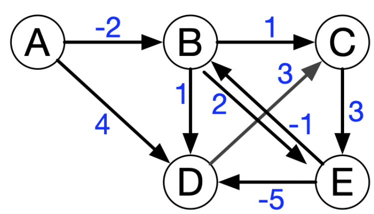


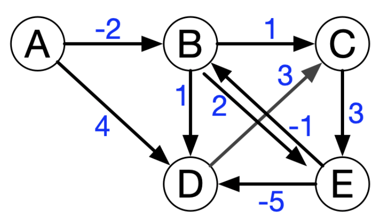


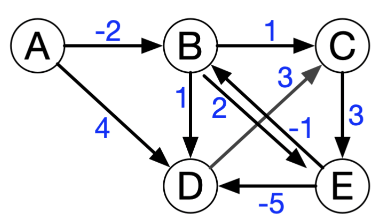






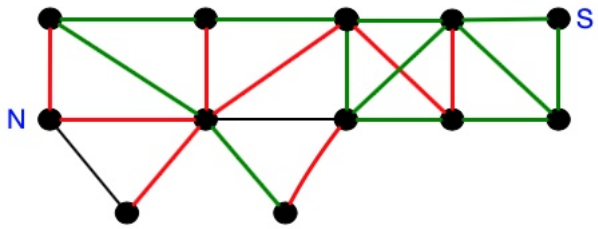






Question 2:

(a) 4 points Let G be an **undirected, unweighted** graph with vertex set V and edge set E. Suppose that each edge e has additional attribute $e.color$, which is either *red* or *green*. A particular vertex is labelled N for the north pole, and a particular vertex is labelled S for the south pole. We would like a path from S to T that is either of **all one color**, OR starts in RED and switches to GREEN. Design an algorithm that solves this problem and justify the runtime of $O(V + E)$. You do not need to write the pseudo-code, but you must carefully describe your steps. Remember to justify the runtime.



(b) 4 points. Consider the same graph as in problem (a), but now assume that the graph is **directed**. Write the steps of the algorithm that solves the problem in this case.

Question 3

8 points

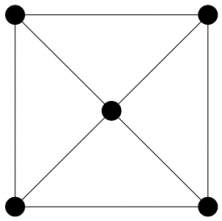
Consider again the graph from problem 3, which is **undirected, unweighted, and connected**. The goal is to find a path of ALTERNATING EDGE COLORS that starts at N and finishes at S. Suppose for simplicity that the path **must start with a green edge**. Your job is the write the pseudo-code for an algorithm that solves this problem. Your solution should be based on DFS-visit from class, and may include an extra parameter for edge color. Do not include any extra data structures.

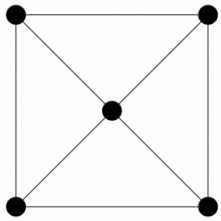
Hint: you will need TWO different v.visited attributes

Question 4

4 points

Consider the graph below. Your job is to place the following weights $1, 1, 2, 2, 3, 3, 4, 4$ on the graph in **two different ways:** in the first way, ensure that only ONE MST exists, and in the second way, place the weights so that more than one MST exists. Justify each of your cases.

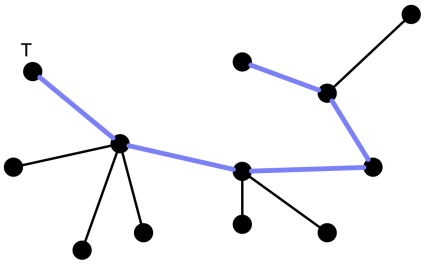




Question 5

8 points

Let G be a connected tree, which as you recall, is a graph without cycles. Note that it is not necessarily a binary tree. Let T be the root of the tree. Update DFS-visit from class so that it returns the length of the **longest simple path** in G from vertex T. Call your procedure LongPath(T). In the example below, the procedure returns 5.



Do not call any external procedures or use any external variables other than those defined in G. Justify that your algorithm has a runtime of $O(V + E)$.

Question 6

6 points

Let G be a directed, weighted graph with **no cycles**. A student tries to update the pseudo-code to Dijkstra's algorithm so that if find the **longest path** from vertex S to all other vertices in G. The update uses a MAX priority queue instead of a MIN priority queue, and extracts the Max instead of the Min.

```
For all vertices v in G, set v.d =-INF
Set s.distance = 0 and s.visited = TRUE
Add s to the Max Priority Queue.
 while Q ≠ empty
    u = Extract-max(Q)
    for each v in Adj[u]
       if v.d <  u.d + w(u, v)
          Increase-Key(Q, v, u.d + w(u, v))
          v.parent = u
```

Does this procedure correctly set the maximum path length from vertex S to all other vertices in G? Justify your answer.

Question 7

8 points

Let G be a weighted, directed graph where each vertex represents a location, and each edge represents a directed edge between two locations. Suppose you are at location M and your friend is at location F. The goal is to somehow **get pizza at location** P and **beer at location** B and then both end up at location W **for a wedding**. The problem is, there are several ways to get there as fast as possible! You could pick up the pizza *and* beer and drive to the wedding and your friend drives directly to the party. Another option would be that you pick up the beer, and your friend picks up the pizza, and then you both go to the party. The goal is to minimise the time it takes for the last person to arrive at the party. For example, if it takes you 10 minutes to get beer and pizza and get to the party, but it takes 15 minutes for your friend to get to the party, the "overall" completion time was 15 minutes.

Design an algorithm that determines the fastest possible option for you both the arrive at the party, including the time to complete the two tasks. Carefully describe any algorithms used from class, their input and outputs, and runtime, and any attributes you indeed to use in the vertex objects. You may assume for simplicity that if one person makes both pickups, the pizza is picked up last, so that it doesn't get cold! Justify the runtime of $O(E \log V)$.

Question 8

6 points

Suppose graph G is a directed, unweighted graph that is KNOWN to have exactly **five** strongly connected components. Your job is to determine how to add **exactly 5 edges** so that there is only ONE strongly connected component. Your job: Design an algorithm that solves the problem and carefully specify each step. You may write things like "run DFS(G)" or "run DFS-visit(v)", but you can't simply state "find the SCCs", since we did not see a procedure with this name in class. The output of your procedure must be the PAIRS OF VERTICES between which you add the edges. Justify the overall runtime of $O(V + E)$.

Question 9

12 points

For each problem below, determine if there is a known polynomial-time algorithm for solving the problem. You must justify your answer using problems and algorithms defined in this course.

1. A set of n children are such that certain pairs of children are friends. A friendship is a bi-directional relationship between two children. The school teacher would like to select at least k children who are all friends with each other.

2. Graph G is a simple, connected, undirected graph. There are n vertices and m edges. The goal is to find at least k vertices and color them pink such that no two pink vertices share an edge.

3. Graph G is a simple, connected, undirected graph. There are n vertices and m edges. The goal is to determine if the vertices can be colored in **two different colors** so that no edge exists between vertices of the same color.

4. A set of m children go shopping in a shop with n different toys. Each child picks exactly TWO of their favourite toys. Note that some children may chose the same toy. The problem is to select at least k toys so that no child goes home with TWO toys (going home with none or one is okay).

Question 10

12 points

Let G be a directed unweighted graph on n vertices, where each vertex is colored either black or white. The goal is to find a a simple cycle through at least k vertices where all vertices are of the same color. Show that this problem is NP complete, using a reduction from a problem from class.